**1. Gradual Increment Method**

*These different methods provide various approaches to smoothly and slowly move your servos to the desired angle. Adjust the steps and delay parameters to further fine-tune the motion to your needs.*

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def slow\_move(servo, start, end, step=1, delay=0.05):

if start < end:

for angle in range(start, end + 1, step):

set\_angle(servo, angle)

time.sleep(delay)

else:

for angle in range(start, end - 1, -step):

set\_angle(servo, angle)

time.sleep(delay)

try:

slow\_move(servo1, 0, 90)

slow\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**2. Linear Interpolation Method**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def linear\_move(servo, start, end, steps=100, delay=0.05):

step\_size = (end - start) / steps

for i in range(steps + 1):

angle = start + step\_size \* i

set\_angle(servo, angle)

time.sleep(delay)

try:

linear\_move(servo1, 0, 90)

linear\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**3. Exponential Decay Method**

import RPi.GPIO as GPIO

import time

import math

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def exponential\_move(servo, start, end, steps=100, delay=0.05):

step\_size = (end - start) / steps

for i in range(steps + 1):

angle = start + (step\_size \* math.exp(i / steps))

set\_angle(servo, angle)

time.sleep(delay)

try:

exponential\_move(servo1, 0, 90)

exponential\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**4. Sinusoidal Motion Method**

import RPi.GPIO as GPIO

import time

import math

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def sinusoidal\_move(servo, start, end, steps=100, delay=0.05):

for i in range(steps + 1):

angle = start + (end - start) \* (1 - math.cos(math.pi \* i / steps)) / 2

set\_angle(servo, angle)

time.sleep(delay)

try:

sinusoidal\_move(servo1, 0, 90)

sinusoidal\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**5. Slow Ramp-Up Method**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def ramp\_up\_move(servo, start, end, steps=100, delay=0.05):

for i in range(steps + 1):

angle = start + (end - start) \* (i / steps)

set\_angle(servo, angle)

time.sleep(delay \* (1 + i / steps))

try:

ramp\_up\_move(servo1, 0, 90)

ramp\_up\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**6. Quadratic Easing Method**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def quadratic\_easing\_move(servo, start, end, steps=100, delay=0.05):

for i in range(steps + 1):

angle = start + (end - start) \* (i / steps) \*\* 2

set\_angle(servo, angle)

time.sleep(delay)

try:

quadratic\_easing\_move(servo1, 0, 90)

quadratic\_easing\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**7. Cubic Easing Method**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def cubic\_easing\_move(servo, start, end, steps=100, delay=0.05):

for i in range(steps + 1):

angle = start + (end - start) \* (i / steps) \*\* 3

set\_angle(servo, angle)

time.sleep(delay)

try:

cubic\_easing\_move(servo1, 0, 90)

cubic\_easing\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**8. Sigmoid Function Method**

import RPi.GPIO as GPIO

import time

import math

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def sigmoid\_move(servo, start, end, steps=100, delay=0.05):

for i in range(steps + 1):

progress = i / steps

angle = start + (end - start) / (1 + math.exp(-12 \* (progress - 0.5)))

set\_angle(servo, angle)

time.sleep(delay)

try:

sigmoid\_move(servo1, 0, 90)

sigmoid\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**9. Logarithmic Function Method**

import RPi.GPIO as GPIO

import time

import math

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def logarithmic\_move(servo, start, end, steps=100, delay=0.05):

for i in range(steps + 1):

progress = i / steps

angle = start + (end - start) \* math.log10(9 \* progress + 1)

set\_angle(servo, angle)

time.sleep(delay)

try:

logarithmic\_move(servo1, 0, 90)

logarithmic\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()

**10. Random Walk Method**

import RPi.GPIO as GPIO

import time

import random

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(13, GPIO.OUT)

servo1 = GPIO.PWM(11, 50)

servo2 = GPIO.PWM(13, 50)

servo1.start(0)

servo2.start(0)

def set\_angle(servo, angle):

duty = angle / 18 + 2

GPIO.output(11, True)

servo.ChangeDutyCycle(duty)

time.sleep(0.5)

GPIO.output(11, False)

servo.ChangeDutyCycle(0)

def random\_walk\_move(servo, start, end, steps=100, delay=0.05):

current\_angle = start

for \_ in range(steps):

step = random.uniform(-1, 1)

current\_angle += step

if start <= current\_angle <= end:

set\_angle(servo, current\_angle)

time.sleep(delay)

try:

random\_walk\_move(servo1, 0, 90)

random\_walk\_move(servo2, 0, 90)

finally:

servo1.stop()

servo2.stop()

GPIO.cleanup()